```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

/*
Dados do problema

Tipo Massa(Kg) Valor
1        3      40
2        5      100
3        2      50
4        4      60
5        6      110

*/


#define num_ind 100 // numero de indivíduos na população
#define taxa_mutacao 20 // taxa de mutação em %
#define limitePeso 98 // capacidade máxima de peso da mochila

#define disponibilidade_1 10 // quantidade disponível do ítem 1
#define disponibilidade_2 12 // quantidade disponível do ítem 2
#define disponibilidade_3 11 // quantidade disponível do ítem 3
#define disponibilidade_4 8 // quantidade disponível do ítem 4
#define disponibilidade_5 7 //quantidade disponível do ítem 5
int populacao[num_ind][5]; //[tipo 1] [tipo 2] [tipo 3] [tipo 4] [tipo 5]
int novaPopulacao[num_ind][5]; //armazena a geração seguinte durante a
reprodução
int fitness[num_ind];
// armazena o fitness de cada indivíduo da população
int fitnessTotal; // armazena a soma de todos os fitness da população para usar
na
seleção
int melhor[5]; // armazena o melhor indivíduo
double aleatorio(){
    return((double)rand()/RAND_MAX);
}

int inteiroAleatorio(int maximo){
    return(rand() % (maximo+1));
}

void calculaFitness(){
    int cont;
    fitnessTotal=0;

    for(cont=0; cont<num_ind; cont++){
        fitness[cont]=(populacao[cont][0]*40)+(populacao[cont][1]*100)+
(populacao[cont][2]*50)+(populacao[cont][3]*60)+(populacao[cont][4]*110);
        fitnessTotal=fitnessTotal+fitness[cont];
    }
}

int calculaPeso(int ind){
    return((populacao[ind][0]*3)+(populacao[ind][1]*5)+(populacao[ind]
[2]*2)+populacao[ind][3]*4)+populacao[ind][4]*6));
}

void inicializaPopulacao(){
    int cont=0;
    while(cont<num_ind){

    populacao[cont][0]=inteiroAleatorio(disponibilidade_1);// determina
```

```
quantidade tipo 1
    populacao[cont][1]=inteiroAleatorio(disponibilidade_2);// determina
quantidade tipo 2
    populacao[cont][2]=inteiroAleatorio(disponibilidade_3);// determina
quantidade tipo 3
    populacao[cont][3]=inteiroAleatorio(disponibilidade_4);// determina
quantidade tipo 4
    populacao[cont][4]=inteiroAleatorio(disponibilidade_5);// determina
quantidade tipo 5
    // verifica a restrição de peso
    if(calculaPeso(cont)<=limitePeso) cont++;
    }
}

int selecionaTorneio(){
    int indiv1,indiv2;
    // seleciona 2 indivíduos aleatoriamente
    indiv1=inteiroAleatorio(num_ind);
    indiv2=inteiroAleatorio(num_ind);

    // retorna o melhor
    if(fitness[indiv1]>fitness[indiv2]){
    return(indiv1);
    }
    else{
        return(indiv2);
    }
}

int selecionaRoleta(){
    int cont=0;
    double roleta=aleatorio()*fitnessTotal; // aponta para um valor de fitness
entre 0 e fitnessTotal
    double somatorioFitness=fitness[0];

    while(somatorioFitness<roleta){
        cont++;
        somatorioFitness=somatorioFitness+fitness[cont];
    }

    return(cont);
}

void reproducao(){
    int cont=0; //conta os filhos para limitar a população
    int pai, mae;
    int pesoFilho1;
    int pesoFilho2;

    while(cont<num_ind){// repete até concluir a próxima geração
        pesoFilho1=limitePeso+1;
        pesoFilho2=limitePeso+1;
        while(pesoFilho1>limitePeso||pesoFilho2>limitePeso){ // repete até que
os dois filhos sejam válidos limite de peso
            // seleciona o pai e a mãe pela roleta
            pai=selecionaRoleta();
            mae=selecionaRoleta();
            // ou seleciona o pai e a mãe por torneio
            //pai=selecionaTorneio();
            //mae=selecionaTorneio();

            while(pai==mae){ // garante que o pai e a mão não sejam o mesmo
indivíduo
                mae=selecionaRoleta();
```

```
                //mae=selecionaTorneio();
            }

            // faz a operação de crossover de 1 ponto para gerar 2 filhos
            // determina aleatoriamente se o corte é entre os cromossomos 1 e 2
(<0,5) ou2 e 3 (>=0,5)
            if(aleatorio()<0.5){
                // [1]|       pai
                //    |[2][3] mae
                // [1] [2][3] filho 1
                novaPopulacao[cont][0]=populacao[pai][0]; // filho 1 gene 1
                novaPopulacao[cont][1]=populacao[mae][1]; // filho 1 gene 2
                novaPopulacao[cont][2]=populacao[mae][2]; // filho 1 gene 3
                novaPopulacao[cont][3]=populacao[mae][3]; // filho 1 gene 4
                novaPopulacao[cont][4]=populacao[mae][4]; // filho 1 gene 5
                //    |[2][3] pai
                // [1]|       mae
                // [1] [2][3] filho 2
                novaPopulacao[cont+1][0]=populacao[mae][0]; // filho 2 gene 1
                novaPopulacao[cont+1][1]=populacao[pai][1]; // filho 2 gene 2
                novaPopulacao[cont+1][2]=populacao[pai][2]; // filho 2 gene 3
                novaPopulacao[cont+1][3]=populacao[pai][3]; // filho 2 gene 4
                novaPopulacao[cont+1][4]=populacao[pai][4]; // filho 2 gene 5
            }
            else{
                // [1][2]|pai
                //       |[3] mae
                // [1][2] [3] filho 1
                novaPopulacao[cont][0]=populacao[pai][0]; // filho 1 gene 1
                novaPopulacao[cont][1]=populacao[pai][1]; // filho 1 gene 2
                novaPopulacao[cont][2]=populacao[mae][2]; // filho 1 gene 3
                novaPopulacao[cont][3]=populacao[mae][3]; // filho 1 gene 4
                novaPopulacao[cont][4]=populacao[mae][4]; // filho 1 gene 5
                //       |[3] pai
                // [1][2]|    mae
                // [1][2] [3] filho 2
                novaPopulacao[cont+1][0]=populacao[mae][0]; // filho 2 gene 1
                novaPopulacao[cont+1][1]=populacao[mae][1]; // filho 2 gene 2
                novaPopulacao[cont+1][2]=populacao[pai][2]; // filho 2 gene 3
                novaPopulacao[cont+1][3]=populacao[pai][3]; // filho 2 gene 4
                novaPopulacao[cont+1][4]=populacao[pai][4]; // filho 2 gene 5
            }

            // calcula o peso de cada um dos indivíduos gerados
            pesoFilho1=(novaPopulacao[cont][0]*3)+(novaPopulacao[cont][1]*5)+
(novaPopulacao[cont][2]*2)+(novaPopulacao[cont][3]*4)+(novaPopulacao[cont]
[4]*6);
            pesoFilho2=(novaPopulacao[cont+1][0]*3)+(novaPopulacao[cont+1]
[1]*5)+(novaPopulacao[cont+1][2]*2)+(novaPopulacao[cont+1][3]*4)+
(novaPopulacao[cont+1][4]*6);
        }
        cont=cont+2; // avança para os próximos 2 filhos
    }

    // transfere a população nova para a variável da população atual
    for(cont=0; cont <num_ind ;cont++){
        populacao[cont][0]=novaPopulacao[cont][0];
        populacao[cont][1]=novaPopulacao[cont][1];
        populacao[cont][2]=novaPopulacao[cont][2];
        populacao[cont][3]=novaPopulacao[cont][3];
        populacao[cont][4]=novaPopulacao[cont][4];
    }
}
```

```
int encontraMelhor(){
    int cont;
    int melhorIndividuo;
    int melhorFitness=0;
    for(cont=0; cont < num_ind; cont++){
        if(fitness[cont]>melhorFitness){
            melhorFitness=fitness[cont];
            melhorIndividuo=cont;
        }
    }

    melhor[0]=populacao[melhorIndividuo][0];
    melhor[1]=populacao[melhorIndividuo][1];
    melhor[2]=populacao[melhorIndividuo][2];
    melhor[3]=populacao[melhorIndividuo][3];
    melhor[4]=populacao[melhorIndividuo][4];

    return(melhorIndividuo);
}

void mutacao(){
    int cont;
    int temp;
    int individuoMutante[5];

    for(cont=0; cont < num_ind; cont++){
        if((aleatorio()*100)<taxa_mutacao){ // verifica a possibilidade deste
indivíduosofrer mutação
            // faz uma cópia do individui a sofrer mutação
            individuoMutante[0]=populacao[cont][0];
            individuoMutante[1]=populacao[cont][1];
            individuoMutante[2]=populacao[cont][2];
            individuoMutante[3]=populacao[cont][3];
            individuoMutante[4]=populacao[cont][4];
            // verifica qual o gene a sofrer mutação
            temp=inteiroAleatorio(2); // 0 a 2
            // encontra um novo valor para o gene
            if(temp==0) individuoMutante[0]=inteiroAleatorio(disponibilidade_1);
            if(temp==1) individuoMutante[1]=inteiroAleatorio(disponibilidade_2);
            if(temp==2) individuoMutante[2]=inteiroAleatorio(disponibilidade_3);
            if(temp==3) individuoMutante[3]=inteiroAleatorio(disponibilidade_4);
            if(temp==4) individuoMutante[4]=inteiroAleatorio(disponibilidade_5);
            // verifica se o novo indivíduo é válido (Limite de peso)
            if(((individuoMutante[0]*3)+(individuoMutante[1]*5)+
(individuoMutante[2]*2)+(individuoMutante[3]*4)+
(individuoMutante[4]*6))<=limitePeso){
                // se for válido grava o indivíduo mutante na população
                populacao[cont][0]=individuoMutante[0];
                populacao[cont][1]=individuoMutante[1];
                populacao[cont][2]=individuoMutante[2];
                populacao[cont][3]=individuoMutante[3];
                populacao[cont][4]=individuoMutante[4];
            }
        }
    }
}

void elitismo(){
    int elemento;
    elemento=aleatorio()*num_ind;
    populacao[elemento][0]=melhor[0];
    populacao[elemento][1]=melhor[1];
    populacao[elemento][2]=melhor[2];
    populacao[elemento][3]=melhor[3];
```

```
    populacao[elemento][4]=melhor[5];
    calculaFitness();
}

int main(){
    srand( (unsigned)time(NULL) );
    int i;
    int cont;
    int contGeracao=0;
    printf("Pressione CONTRO-C para sair\n");
    inicializaPopulacao();
    calculaFitness();
    printf("Geracao %d \n",contGeracao);

    for(i=0; i <num_ind ;i++){
        printf("%d:-> %d %d %d (%d Kg)(R$%d)\n",i,populacao[i][0],populacao[i]
[1],
        populacao[i][2],calculaPeso(i),fitness[i]);
    }

    i=encontraMelhor();
    printf("Melhor individuo %d Fitness = R$%d\n",i,fitness[i]);
    printf("Fitness total %d\n-----------------------------------------------\n",
    fitnessTotal);
    getchar();

    while(contGeracao<1000){
        contGeracao++;
        //-------------------------------------
        reproducao();
        mutacao();
        calculaFitness();
        elitismo();
        i=encontraMelhor();
        //-------------------------------------
        printf("Geracao %d \n",contGeracao);

        for(cont=0; cont <num_ind ;cont++){
            printf("%d:-> %d %d %d (%d Kg)(R$%d)\n",cont,populacao[cont]
[0],populacao[cont][1],populacao[cont][2],populacao[cont][3],populacao[cont]
[4],calculaPeso(cont),fitness[cont]);
        }

        printf("Melhor individuo %d Fitness = R$%d\n",i,fitness[i]);
        printf("Fitness total %d\n-----------------------------------------------\
n",
        fitnessTotal);
        getchar();
    }
}
```